Organizing Projects 8

Projects are at the heart of the THINK Project Manager. A project keeps track of all your source files and stores information needed to debug, compile, and create your program. This chapter explains how to create a project, add files to it, and read the project window.

What you should know

You should know how the THINK Project Manager works. If you haven't done so, work through Chapter 4, "Tutorial: MiniEdit." That chapter takes you step by step through building an application in the THINK Project Manager and gives you the practical background you need.

Contents

What Is a Project? 123

The Project Window 123

What You Can Put in Your Project 126
- Source files 127
- Libraries 127
- Resources 128
- Notes 129

Creating a New Project 129
- Using project models 129
- Using project stationery pads 131
- Creating your own project models 133

Opening an Existing Project 133

Closing Projects 135

Adding Files to Projects 135
- Adding a file in the THINK Editor 135
- Adding a file on disk 136

Selecting Files. 137

Removing Files from Projects 137

Setting the Project Type 137

Segmenting Projects 139
Letting the THINK Project Manager segment your project . 139
Moving files into segments 140
Moving entire segments 141
Deleting a segment 141
Changing a segment's name and attributes 141

Compressing Projects 142

Recovering Corrupted Projects 142

Setting Project Options. 143
Using the Options dialog 144

Changing File Extensions 146
Changing the translator for an extension 147
Adding an extension 147
Deleting an extension 148

Changing Your Project Window View 148
Viewing your project's components 148
Viewing segment numbers 150

Project Limits 151

What Is a Project?

Most programs consist of several source files, libraries, and resource files. A small program might be made up of one or two source files. A large, full-featured program, though, usually consists of 150 to 300 source files that use a couple of compilers, some libraries, and resource files. In traditional development environments, you create and edit your source files and resource files with a text editor. Then you compile each file with the proper compiler to generate an object file. Finally, when you have all your object files together, the linker links all the object files and libraries to produce an executable application. As your program gets larger, it becomes harder to keep track of the files that need to be recompiled and the compilers they use. And once you've compiled the changed files, you still have to link all the object files together.

The THINK Project Manager is an integrated environment: the editor, compilers, and linker are all part of the same application and can work together. After you edit a source file with the editor, the THINK Project Manager remembers that you changed it. When you run or build your project, the THINK Project Manager decides those files to recompile and those compiler to use. As the compiler generates object code, it also generates information for the linker and debugger. The linker links your files instantly and the debugger lets you watch your code execute line by line. The thing that holds all the pieces together is the project.

The project is at the heart of the THINK Project Manager. It takes over the functions of several other files in more traditional development environments. The project holds the object code of all your compiled source files and maintains the dependencies between them. It keeps track of which files you edited and which compilers it needs to recompile them. The project also keeps track of information used by the source level debugger.

The Project Window

The project window displays information about each file in your program. To tell the THINK Project Manager that a file is part of your program, you add it to your project, and the file name appears in the project window.

Figure 8-1 shows a project window. It lists the names of the files and the sizes of the files' object code. In this project window, none of the files has been compiled, so the Code column shows zeros for the

files. (Since there is overhead for the project, the Totals line isn't zero.)

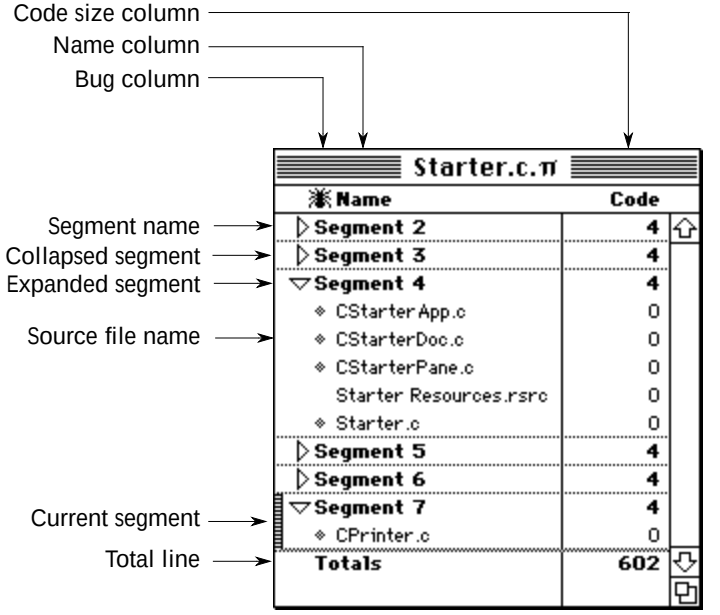


Figure 8-1 The Project window

For more information on segments, see "Segmenting Projects" later in this chapter.

Names in bold are segment names. Segments are units of code that the system can swap in and out of memory. Every file in your project belongs to a segment. Dotted lines separate the segments in the project window. The THINK Project Manager places the next file you add into the current segment, which has gray hatching along the left edge.

The THINK Project Manager lets you collapse and expand segments, much as the Finder lets you collapse and expand folders in list views. A collapsed segment does not list its files and is marked with a right-pointing triangle (▷). An expanded segment lists its files and is marked with a down-pointing triangle (▽). To collapse or expand a segment, click its triangle. To collapse or expand all the segments

in a project, hold down the Option key as you click any segment's triangle.

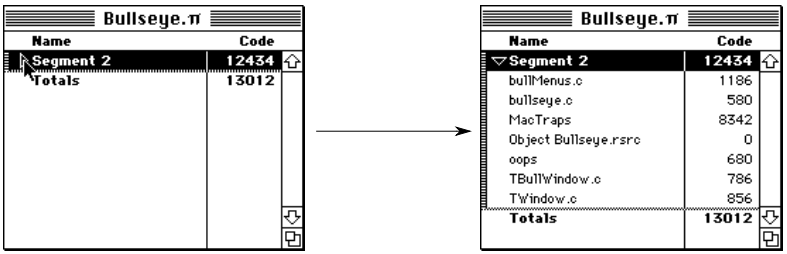


Figure 8-2 Expanding a segment

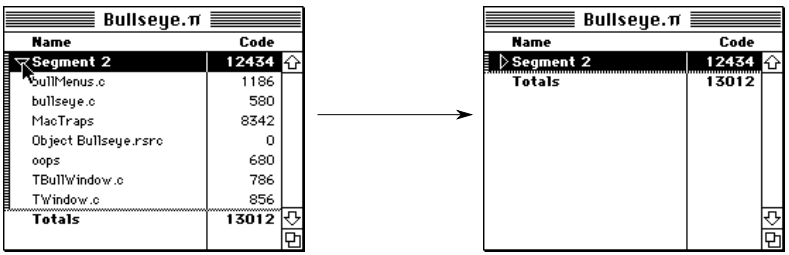


Figure 8-3 Collapsing a segment

To learn about the THINK Editor, see Chapter 10, "Editing Files."

To edit a text file, double-click its name, and the THINK Project Manager opens it in an editor window. If you're running System 7, you can edit other files by double-clicking them too. If it's a project, the THINK Project Manager closes the current project and opens it. Otherwise, the THINK Project Manager asks the Finder to open it with the application that created it.

You can click and drag file names and segment names within the project window. To learn why you'd want to, see "Segmenting Projects" on page 139.

To learn about the THINK Debugger, see Chapter 12, "Debugging Programs."

The bug column (🐞) shows those files that have debugging information stored in your project. This column appears only if the **Use Debugger** command from the **Project** menu is checked.

To display more information in your project window, see "Changing Your Project Window View" later in this chapter.

The Name column shows you the names of the files and segments in your project. The Code column shows you the size of the files' and segments' object code in bytes. When you add a new file to a project, the object size is zero since the file hasn't been compiled

yet. When a file is compiled, the size of its object code appears in the Code column.

Note

Files mapped to translators that don't generate code—such as THINK Rez and the Resource Copier, which only contribute resources—always display 0 (zero) in the Code column.

What You Can Put in Your Project

Your project can contain many types of files: source files, libraries, resources, and notes.

The THINK Project Manager uses translators to compile the different types of files when you run or build your project. They're in the `Translators` folder in your THINK Project Manager folder.

The file's extension tells the THINK Project Manager which translator to use. An extension is a string beginning with a period that's added at the end of a file name. For example, `Bullseye.cpp` has the extension `.cpp` and is a Symantec C++ file. `Bullseye.r` has the extension `.r` and is a THINK Rez file.

Most files must end in an extension that the THINK Project recognizes before you can add them to your project. Project libraries and binary libraries are the only files that don't require them. The THINK Project Manager recognizes the extensions in Table 8-1 by default. To add or change extensions, see "Changing File Extensions" later in this chapter.

File Type	Extension
THINK C source file	`.c` or `.asm`
Symantec C++ source file	`.cp` or `.cpp`
Resource file	`.rsrc`
THINK Rez source file	`.r`
Project library	not necessary
Binary library	not necessary
MPW .o library	`.o`
Note file	`.note`

Table 8-1 Default file extensions

Source files

Source files contain code that a translator turns into object code or resources. Symantec has three source code translators for the THINK Project Manager: THINK C, THINK Rez, and Symantec C++. By default, THINK C source files end in `.c` or `.asm`, THINK Rez files end in `.r`, Symantec C++ source files end in `.cpp` or `.cp`.

Use the `.asm` *ending when you need to combine inline assembly and C++ source code in the same project. For more information, see the* C++ Compiler Guide.

When you run or build your project, the THINK Project Manager compiles your source files and adds the object code to your project. Unlike other development environments, the THINK Project Manager doesn't generate separate object files for each source file.

Libraries

Libraries contain compiled functions that you can use in many different projects. You can use three types of libraries:

- **Project library**. You can use any THINK Project Manager project that contains only one segment as a library. For example, the ANSI libraries that come with THINK C and Symantec C++ are projects. When you use a project as a library, the smart linker uses only the code it needs. A project library doesn't need an extension for the THINK Project Manager to recognize it. Because it's already compiled, it doesn't require a translator.

- **Binary library**. This is a file created with the **Build Library** command. It's seldom used now and is provided for compatibility with older versions of THINK C and some third-party products. A binary library usually ends in `.lib`, but it doesn't need an extension for the THINK Project Manager to recognize it. Because it's already compiled, it doesn't require a translator.

- **MPW .o library**. This is a library created with the Macintosh Programmer's Workshop (MPW) or THINK Pascal. By default, it ends in `.o`. You cannot edit a `.o` library in the THINK Project Manager. You must use MPW. Because it must be translated from the MPW object format to the THINK Project Manager object format, it requires a translator, the .o Converter.

When you run your project, the THINK Project Manager automatically loads project and binary libraries into it. To learn how to load MPW `.o` libraries, see Chapter 26, "Using Object Files."

Resources

Most Macintosh applications use resources for menus, window templates, dialogs, and so on. Two types of files add resources to your project:

- **Resource files**, which you create with a resource editor, such as ResEdit or Resorcerer. The file's name must end in `.rsrc`.
- **Rez files**, which contain text descriptions of resources. The THINK Rez translator converts these descriptions into resources. The THINK Rez language is described in Chapter 27, "Using THINK Rez." The file's name must end in `.r`.

You can have as many resource-producing files in your project as you want. Although resource files appear in segments in the project window, they don't add any code to them and display zero in the Code size column even after they've been compiled.

Resources aren't stored in project files since project files have resources of their own, and you could accidentally overwrite those resources.

When you run or build your project, the THINK Project Manager copies the resources from these files into the project resource file, which contains all the resources for your project. If you're running the program, the THINK Project Manager then opens the project resource file so you can access its resources. If you're building the project, the THINK Project Manager copies the project resource file into your finished program.

The THINK Project Manager automatically maintains the project resource file. It is in the same folder as your project, and its name is your project's name with `.rsrc` added to the end. For example, if the name of your project is `MyProject`, the project resource file is `MyProject.rsrc`.

*For more information on **Remove Objects**, see "Compressing Projects" and "Recovering Corrupted Projects", later in this chapter.*

The THINK Project Manager keeps track of which file a resource comes from. If you delete, add, or update a resource in a resource-producing file, the THINK Project Manager deletes, adds, or updates that resource in the project resource file. If resources in two different files have the same type and ID number, the THINK Project Manager displays an error message. If you choose **Remove Objects** from the **Project** menu, it removes the resources in the project resource file that came from resource files or Rez files.

Notes

The THINK Project Manager lets you add any other type of document to your project. These documents are called note files, since they usually contain notes about your program. For example, you could add

- A text file your application reads from
- A MORE file listing suggestions
- A Microsoft Word file documenting the interface, and
- A MacDraw file illustrating possible splash screens.

By default, a note file ends in `.note`. Its translator is `<<none>>`. The translator just ignores a note file when you build or run your program. Although note files appear in segments in your project window, they don't add any code or resources to your project.

Creating a New Project

The THINK Project Manager gives you two ways to create new projects. Usually, you'll want to create one from a project model, which includes many of the files and libraries you'll need in the project. If you're running under System 7, you can also create a project from a project stationery pad.

Using project models

For more information on how to create your own project models, see "Creating your own project models" later in this chapter.

Usually when you create a new project, you'll create it from a project model, which defines what source files, resource files, and libraries your project will need. When your create a new project, the THINK Project Manager creates a new project document, some source files for you to edit, and resource files. The project document contains many of the source files, resource files, and libraries you'll need.

If you have Symantec C++ for Macintosh, you'll see more project models in this list.

When you choose the **New Project** command from the **File** menu, you see the dialog in Figure 8-4. The dialog contains a list of the available project models. If the Create folder option is on, the THINK Project Manager creates a new folder to put the new project document and other files in. Otherwise, it puts the files in the current folder.

To choose a project model, double-click it, or select it and click Create.

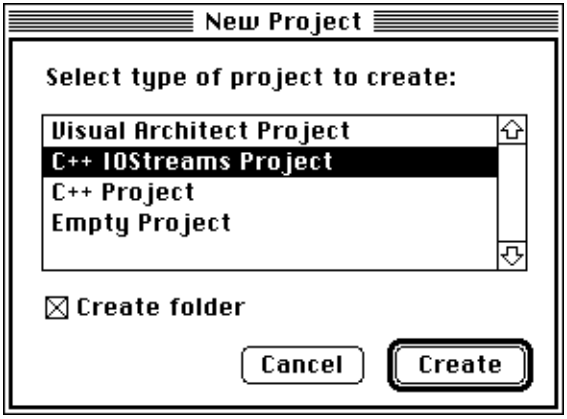


Figure 8-4 New Project dialog

Next, you see a standard file dialog, as in Figure 8-5. Enter a name for the project and click Save.

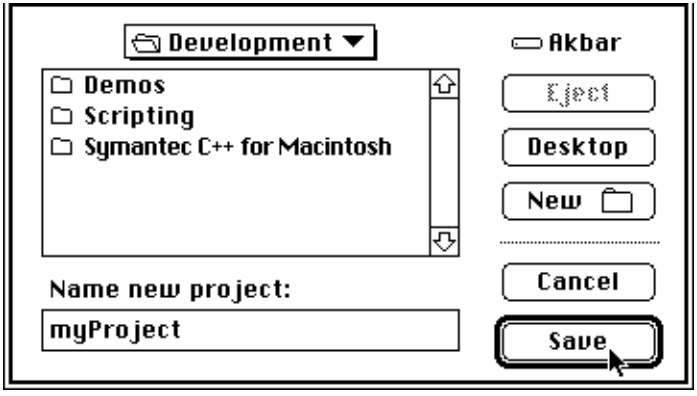


Figure 8-5 Naming the project

Note

Don't use a name that ends in .π. The THINK Project Manager will add a suffix to the name of the project file for you.

The THINK Project Manager creates a folder with your project name and the suffix ƒ. The folder contains files you'll edit and a project

document with your project name and the suffix .π, as shown in Figure 8-6. The THINK Project Manager displays the project window. Figure 8-7 shows the project window and a file for you to edit.

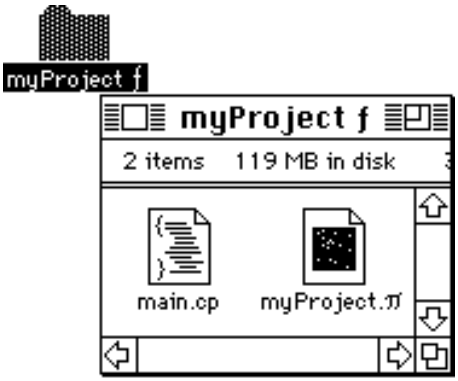


Figure 8-6 New project folder

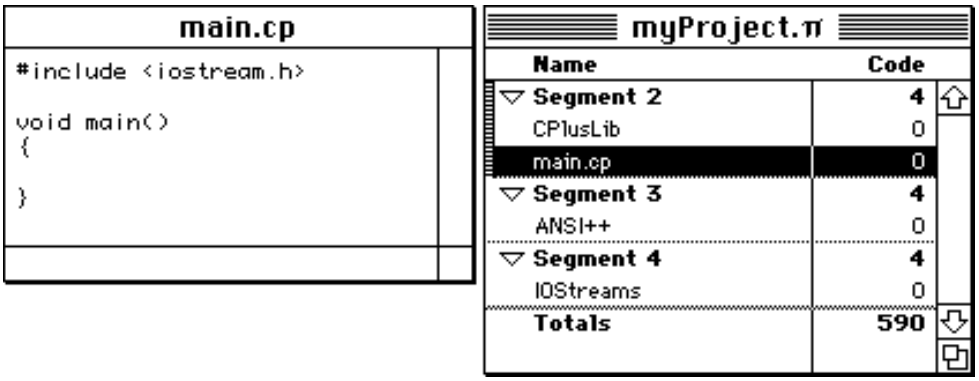


Figure 8-7 New project window and edit window

Using project stationery pads

To create a new project from a stationery pad, use the **Open Project** command in the **File** menu. Select the stationery project you want from the standard file dialog that appears. You see another dialog, as

in Figure 8-8. Move to the folder where you want to put the new project.

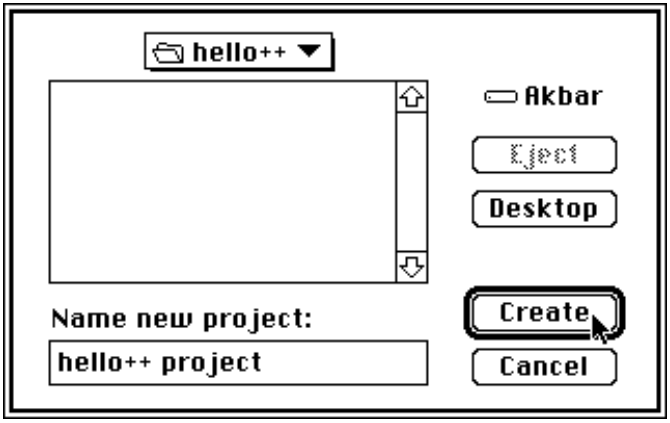


Figure 8-8 Naming a new project

To make a "π", type Option-P.

To name the new project file, type in the name, and click Save. By convention, project names end either in the word "project" or with a ".π" suffix. You don't have to name your projects this way.

The THINK Project Manager makes a copy of the project stationery pad and displays a project window.

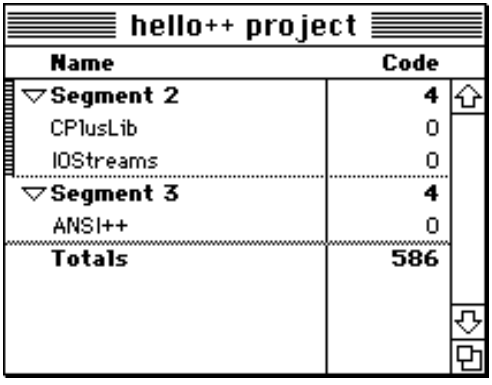


Figure 8-9 New project from a stationery pad

Note

To create a project stationery Pad, create a project in the THINK Project Manager, and choose **Remove Objects** from the **Project** menu to remove its object code. Go to the Finder, select the project, choose **Get Info**, and click the Stationery Pad option.

Creating your own project models

You can create your own project models. If you frequently create lots of projects that contain the same libraries and similar source files, creating your own project model helps you get started programming faster.

To create a project model, follow these steps:

1. Make a new folder in the `(Project Models)` folder, which is in the folder that contains the THINK Project Manager.

2. Name the folder. The folder's name appears as the model's name in the **New Project** dialog. If you want this project model to be the first one listed in the **New Project** dialog, put a • (Option-8) in front of its name.

3. Put a project document in the folder. Create the project in the THINK Project Manager, add the source files and libraries you want, and choose **Remove Objects** from the **Project** menu to remove its object code.

4. Name the project document. To include the name that the user enters in the **New Project** dialog, put `@1` in the name. For example, if the project model's project document is named `@1.π` and the user names the project `MyProject`, the user's project document is named `MyProject.π`.

5. Put the rest of the files in the folder.

Opening an Existing Project

To open an existing project, you can double-click its icon from the Finder, or you can use the **Open Project** command from the **File** menu.

Note

You can have only one project document open at a time in the THINK Project Manager.

If you already have a project open, there are several ways to open a different one. If you're using System 7, you can open it from the Finder by double-clicking it. The THINK Project Manager closes the old project and opens the new one. If the project is a library in the currently open project, you can double-click its name in the project window. The THINK Project Manager closes the old project and opens the library's project. You can also choose **Close Project** from the **Project** menu. The THINK Project Manager closes the current project and displays a standard file dialog that lets you open a different one.

The quickest way to open a project is to use the **Switch to Project** hierarchical menu in the **Project** menu. This menu lists all the projects you've opened since you started the THINK Project Manager. To open a project, select it from the menu. If you're running System 7, you can add more projects to this menu by putting an alias to the project in the `Projects` folder, which is in the folder that contains the THINK Project Manager.

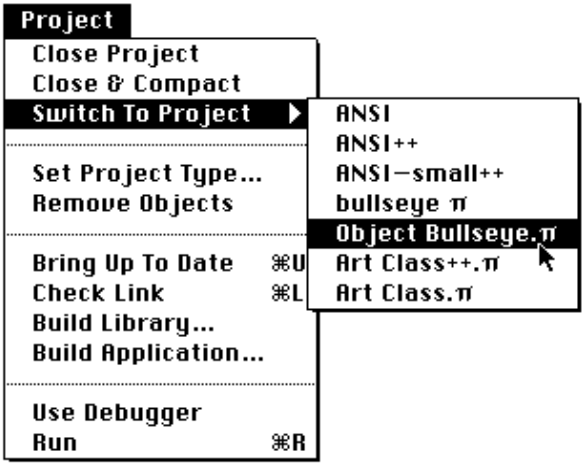


Figure 8-10 The Switch to Project hierarchical menu

Note

Usually, the THINK Project Manager asks you to open a project when you first start it or when you close a project. To avoid this dialog, turn on the Don't ask for project on open option on the Preferences page of the **THINK Project Manager options** dialog.

Closing Projects

Use the **Close Project** command in the **Project** menu to close the project document. The THINK Project Manager closes any open source files that are part of the project. If you've edited a file and haven't saved it, the editor asks if you want to save the changes before closing the project document.

After you close a project, the THINK Project Manager displays a standard file dialog that lets you open another project. If you don't want to open a project, click the Cancel button.

If you don't want the THINK Project Manager to ask you to open a project after it closes one, select the Don't ask for project on open option on the Preferences page of the **THINK Project Manager options** dialog.

Adding Files to Projects

The THINK Project Manager gives you two ways to add files to a project.

Adding a file in the THINK Editor

If you're editing a file with the THINK Editor that isn't in the Project window, use the **Add** command in the **Source** menu to add the file in the current Edit window to the project. The name of the file is in the command name. For example, if the file is `utils.c`, the command is **Add 'utils.c'**. If it's a new file, you must save it with **Save As** before you can add it to the project.

Adding a file on disk

To add a library or file on disk to a project, choose the **Add Files** command in the **Source** menu. You see the dialog in Figure 8-11.

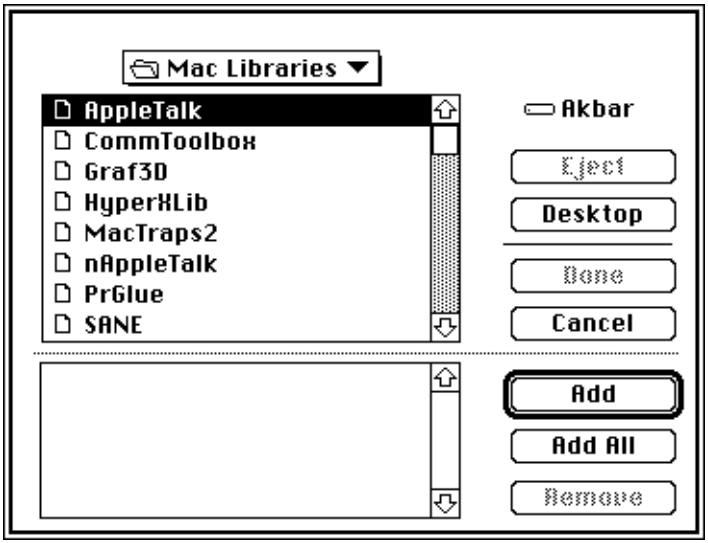


Figure 8-11 Add Files dialog

The top list displays the contents of the folder you're in. It shows only files that end in extensions that the THINK Project Manager recognizes and libraries. The bottom list displays the files that will be added to your project when you click Done. You add files to the bottom list with these commands:

- To add a file, select it and click Add, or double-click it.
- To add all the files from the folder you're in, click Add All.
- To open a folder, select it and click Open, or double-click it. (Add becomes Open when you select a folder.)
- To remove a file from the bottom list, select it and click Remove.
- When you finish selecting files to add, click Done.
- If you change your mind and don't want to add any files, click Cancel.

When you add a library to a project, its object code isn't loaded, so its object code size is zero. You can use the **Load Library** command in the **Source** menu to load a library, or you can let the THINK Project Manager's auto-make facility load it. The auto-make facility loads a library if it was never loaded or if it changed since the last time it was loaded.

Selecting Files

The THINK Project Manager gives you two ways to select files in your project: by clicking the mouse and by typing. To select a file with the mouse, click its name.

To select a file by typing, make sure the project window is the active window and type the first few characters of the file name. The THINK Project Manager selects the first displayed file name that matches what you typed. It doesn't look for matches in collapsed segments. If what you typed doesn't match a displayed file name, any file that might be selected is deselected. When what you type matches more than one displayed file, you can use the Tab key to cycle among all the names that match. Also, you can use the Up or Down Arrow keys to select the file above or below the currently selected file.

Note

If your keyboard doesn't have arrow keys, use the Backspace key. In the project window, use Backspace as you would Up Arrow, and Shift-Backspace to mean Down Arrow.

Removing Files from Projects

To remove a file from a project, select a file in the project window and choose the **Remove** command from the **Source** menu. This command just removes the file from the project. It doesn't delete the file from your disk, so if you remove a file accidentally, you can add it back.

The THINK Project Manager also lets you use projects as libraries. For more information on how to create a library, see "Building Libraries" in Chapter 13, "Building Projects."

Setting the Project Type

The THINK Project Manager lets you build four kinds of projects: applications, desk accessories, device drivers, and code resources. To set the project type, use the **Set Project Type** command in the **Project** menu. This command displays a dialog box that lets you set

type-specific attributes for your project. For every project you can set the type and creator of the final file.

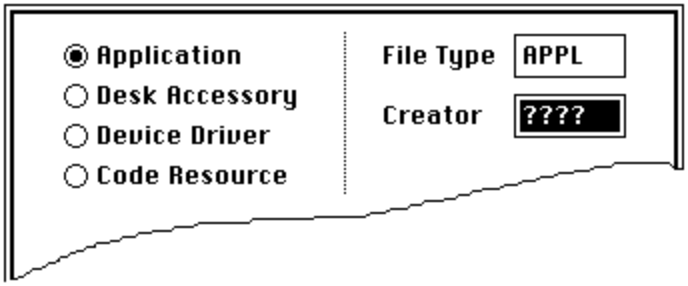


Figure 8-12 The file type and creator

Be sure to set the project type before you compile the project. You can change project types as you wish, but you have to recompile the whole project if you do so. In Chapter 13, “Building Projects,” a section describes the **Set Project Type** options and how to build each type of project.

To learn about File Types and Creators (also called signatures), see Inside Macintosh: Macintosh Toolbox Essentials, *Chapter 7, “Finder Interface.”*

The File Type and Creator of a file let the Finder know what icon to display and what application to open it with. The Finder treats some types, like `'RDEV'`, `'INIT'`, and `'cdev'`, specially.

When you set the project type to something other than **Application**, the name of the **Build Application** command in the **Project** menu changes accordingly. For example, if you set the project type to **Desk Accessory**, the command reads **Build Desk Accessory**.

Changing the type of a project changes the way a project is built, not the way a program behaves. You can’t turn an application into a desk accessory merely by changing the project type. Desk accessories are structurally different from applications. The same applies for the other types.

When you choose one of the **Build** commands (**Build Application**, **Build Desk Accessory**, and so on.) from the **Project** menu, the THINK Project Manager creates a file and creates the `'CODE'` resources for applications, a `'DRVR'` resource for desk accessories and drivers, or the type of resource you specify for your code resource. The THINK Project Manager also creates the resources it uses to manage global data and inter-segment calls.

Segmenting Projects

To learn more about segments, see Inside Macintosh: Processes, *Chapter 7, "Segment Manager."*

Most Macintosh programs are made up of several segments. Segments are units of object code that can be swapped in and out of memory as needed. The Macintosh Operating System limits segments to 32K, so if you're writing a large program, you must segment your code.

The THINK Project Manager lets you segment not only applications, but also desk accessories, device drivers, and code resources. Applications can have up to 254 segments. Desk accessories, device drivers, and code resources can have up to 30 segments.

Letting the THINK Project Manager segment your project

If your program contains a segment that is over 32K, the THINK Project Manager displays a dialog asking if you want it to segment your project automatically, as shown in Figure 8-13.

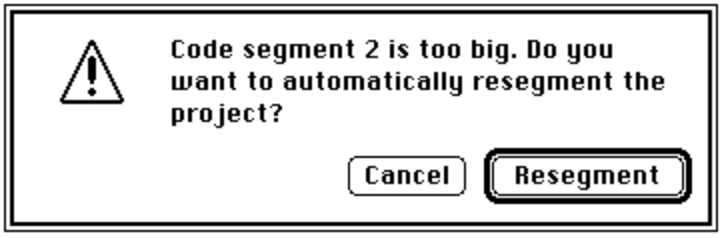


Figure 8-13 Automatic resegment prompt

If you click Resegment, the THINK Project Manager moves enough of your files into new segments so that there is no segment larger than 32K. If you click Cancel, the THINK Project Manager doesn't change your project, and you must resegment it yourself before you can run it.

To have the THINK Project Manager segment your projects without asking you first, select the Always resegment on overflow option on the Preferences page of the **THINK Project Manager options** dialog.

Moving files into segments

Dotted lines separate segments, and the current segment has hatching running along the left edge. When you add files to your project, they're added to the current segment.

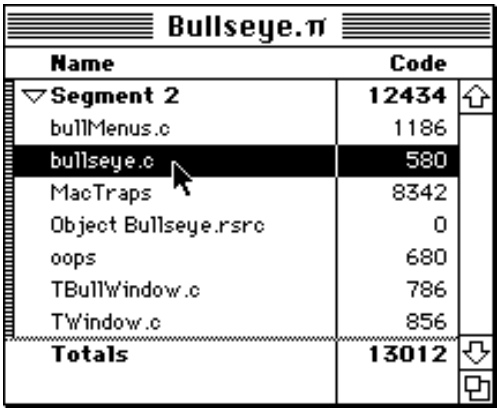


Figure 8-14 A project with one segment

To move a file into a segment, click its name and drag it into the segment. You can move files into collapsed, as well as expanded, segments. To create a new segment, click a file name and drag it below the Totals line If you're segmenting a project with more files than fit in the project window, the window scrolls as you drag the file near the top or bottom of the window.

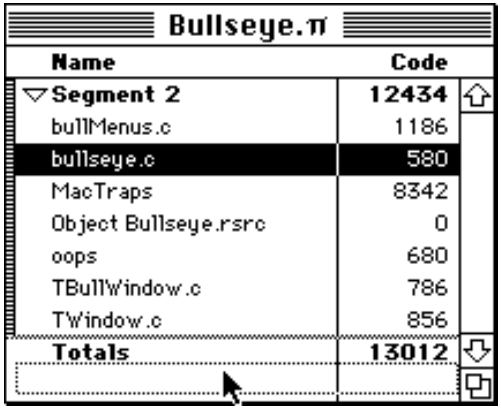


Figure 8-15 Moving a file to a new segment

Since segments provide a handy way to organize your files within a project, you might want to create several small segments of related files, even if your program fits into one or two large segments.

Moving entire segments

If you want to rearrange whole segments at a time, click a segment name and drag it. The entire segment is placed before the segment where you stop dragging. Moving segments affects only the project window display; it doesn't change the segment numbering.

Deleting a segment

To delete a segment, move all the files from it into other segments. The THINK Project Manager automatically gets rid of a segment when you move the last file out of it. When you use the **Add Files** command, new files are placed in the current segment.

Changing a segment's name and attributes

To change a segment's name and attributes, double-click the segment name. The dialog in Figure 8-16 opens.

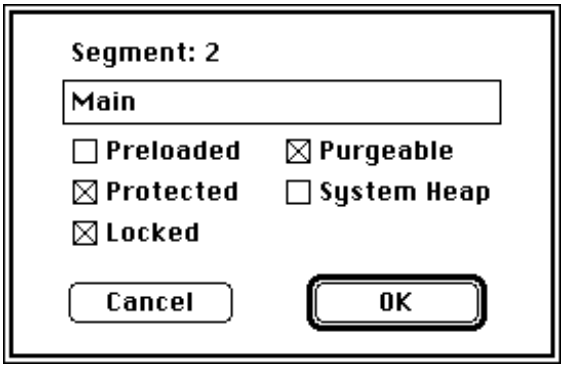


Figure 8-16 Segment attribute dialog

For more information on these attributes, see Inside Macintosh: More Macintosh Toolbox, *"Resource Manager."*

At the top of the dialog is the number that the THINK Project Manager assigned the segment. In the middle is a text-edit box where you can edit the segment name. If the segment has no name, the project window displays the segment's number. At the bottom are checkboxes that let you change the segment's attributes. In applications, Protected, Locked, and Purgeable are checked by default. In desk accessories, device drivers, and code resources, Purgeable is checked by default.

Compressing Projects

Project files can grow to be large and can quickly fill up a hard disk. The THINK Project Manager gives you many ways to compress your projects to save space.

The easiest way is to have the THINK Project Manager store your debugging information in separate files and not in your project document. Just turn on the Store debug info separately option on the Debugging page of the **THINK Project Manager options** dialog. For more information, see "Store debug info separately" in Chapter 12, "Debugging Programs."

Another way is to choose **Close & Compact** from the **Project** menu. This command is useful if you're creating a library, archiving your project for a short time, or sending a small project to someone with a modem. **Close & Compact** compresses your project as much as possible without removing any object code. It then closes your project and lets you open a new one, like the **Close Project** command described in "Closing Projects" later in this chapter. If you want the THINK Project Manager to compact your projects every time you close them, choose the Always Compact Projects option on the Preferences page of the **THINK Project Manager options** dialog.

If you don't mind recompiling your project the next time you use it, choose **Remove Objects** from the **Project** menu. This command is useful if you're archiving your project for a long time or sending a large project to someone with a modem. **Remove Objects** removes all the object code from your project, making it as small as possible. It does *not* close your project. The next time you run or build the project, the THINK Project Manger recompiles all its files and reloads all its libraries.

Note

You cannot use a project with the object code removed as a library.

Recovering Corrupted Projects

When you're debugging a program, you may run across errors that you can't explain. Maybe you see an `Illegal Instruction` error you can't trace, or maybe the THINK Project Manager can't open your project.

A common cause of these problems is a corrupted project. When you run a faulty program under the THINK Project Manager, the program can accidentally write to its project file and corrupt it. Writing to an uninitialized pointer or a pointer with the wrong value is frequently the cause. If you have a bug you can't explain or have a project that the THINK Project Manager refuses to open, try these solutions.

- Recompile your project. If you can't compile or run your project, its object code may be corrupted. Choose **Remove Objects** from the **Project** menu and recompile your project.
- Rebuild your project. If you can't open your project, try opening another one. If that one opens, rebuild your project from scratch, creating a new project file and adding the source files.
- Make a backup of your project. If a program corrupts its project frequently, try making a backup of the project. Build the project from scratch and compile it with the **Bring Up To Date** command. When your project is corrupted, make a copy of the backup and open it. Choose the **Make** command, click Use Disk to select the files that you've changed, and click Make to compile the files.
- Use the scripts that come with the THINK Project Manager to create a text description from your project file. When you need to re-create your project, use the scripts that create a project from the text descriptions. The THINK Project Manager comes with scripts for both AppleScript and Frontier.

*For more information on **Make**, see "Compiling files that auto-make misses" in Chapter 11, "Running Programs."*

Setting Project Options

The **Options** command gives you a lot of control over how the THINK Project Manager behaves and how its translators compile your code. This section tells you how to use the **Options** dialog and where to go for more information on what the options do.

Using the Options dialog

When you select **Options** in the **Edit** menu, you see a hierarchical menu with five items (four if you don't have Symantec C++), as shown in Figure 8-17.

Figure 8-17 Options hierarchical menu

The first item is for the THINK Project Manager and lets you determine how it behaves. The other items are for translators and let you determine how they compile code. To set options for one of them, choose its name from the hierarchical menu. You see a dialog like the one in Figure 8-18.

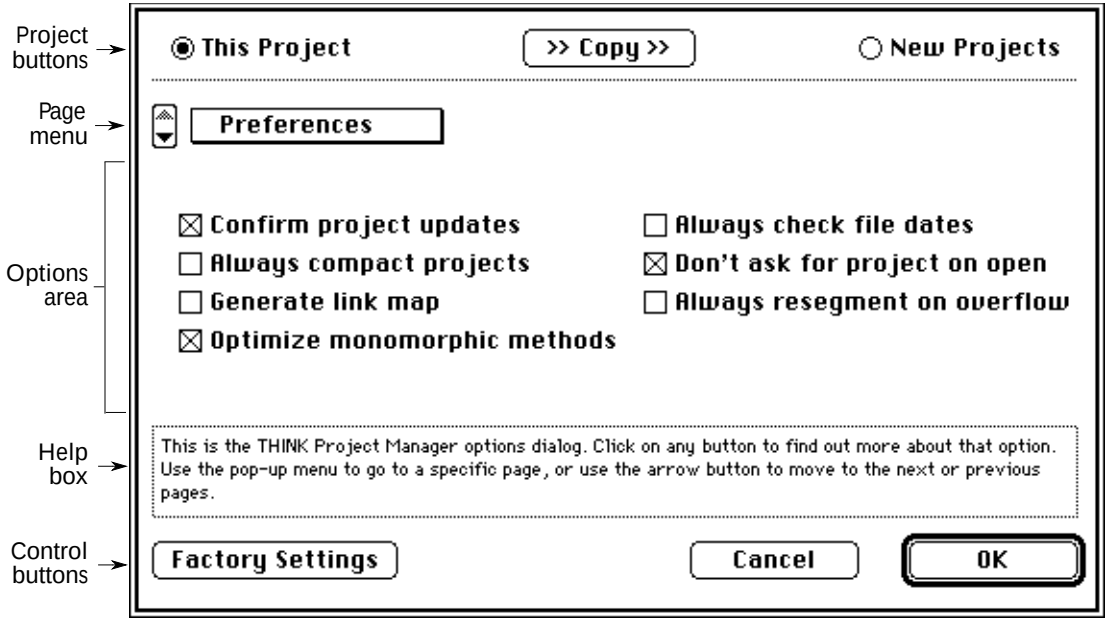


Figure 8-18 THINK Project Manager Options dialog

The Options Area in the middle is where you actually set your options and is different in each page of the dialog. The rest remains the same in all the options dialogs.

The Help Box near the bottom displays information about an option when you click on the option. To get help on an option without changing its setting, click it without releasing the mouse button, move the mouse off the option, and then release the mouse button.

When an options dialog has a lot of options, it has more than one page. The Page Menu in the upper-left corner lets you choose among them. Click the pop-up menu and select the page name. You can also scroll through the different pages with the arrow keys or the arrow buttons to the left of the pop-up menu.

Figure 8-19 Page menu

The Project Buttons on the top let you choose if you're setting the options for the current project or setting the defaults that THINK Project Manager will use when you create a new project. Use the Copy button at the top of the dialog box to copy the defaults to the current project, or if you want the options you've set for a particular project to be the THINK Project Manager options. Note that even though only one page of the options shows up in the dialog at one time, the Copy button copies *all* of the options, even the ones in the pages you can't see. However, it doesn't affect the settings in other option dialogs.

The control buttons are at the bottom. When you click the Factory Settings button, the THINK Project Manager sets all the options on all the pages back to their factory settings, the settings they had when you first installed the THINK Project Manager.

When you click the OK button, the changes for all pages of the dialog are saved. When you click the Cancel button, the changes for all the pages of the dialog are discarded. It's as though you never selected the Options command.

Changing File Extensions

The THINK Project Manager lets you change the translator it uses for an extension and lets you add new extensions. For example, you may want to change the translator for .c files from THINK C to Symantec C++, or add the extension `.c++` to the list of extensions for Symantec C++. You edit the extensions in the Extensions page of the **THINK Project Manager options** dialog, shown in Figure 8-20.

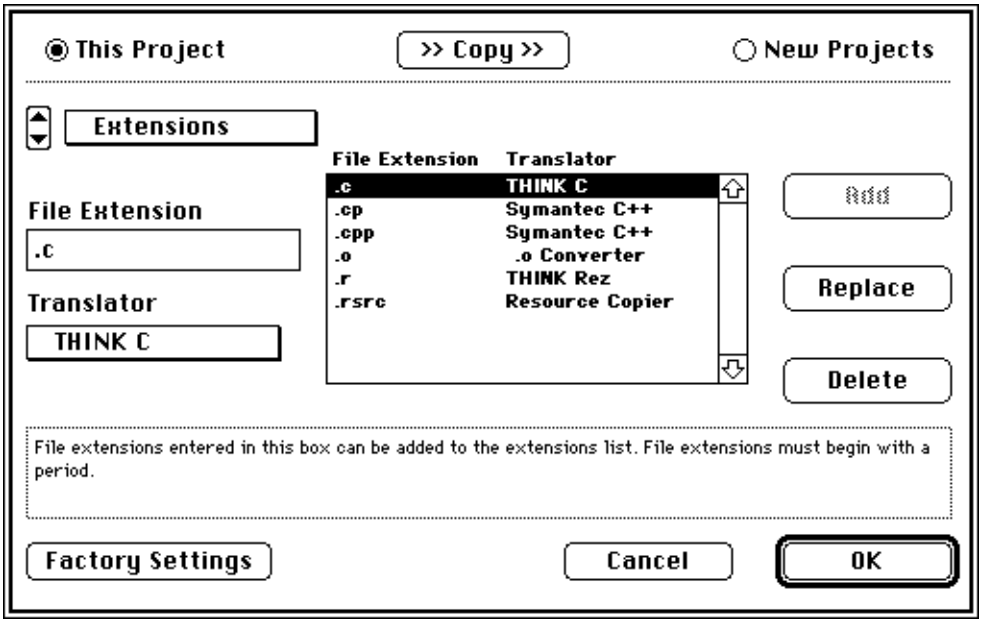


Figure 8-20 Extensions page

In the middle of the dialog is a list of the extensions that the THINK Project Manager recognizes and the translator it uses for each. On the left are a File Extension text box and a Translator pop-up menu (shown in Figure 8-21), in which you edit an extension and choose a

translator for it. On the right are buttons that let you add, replace, or delete the extension you're editing on the left side.

Figure 8-21 Translator menu

Changing the translator for an extension

To change the translator that's used for a particular extension, follow these steps:

1. Select an extension from the list. The extension text appears in the File Extension box and the translator is selected in the Translator menu.

2. Choose a new translator from the Translator menu.

3. Click the Replace button.

Note

If you click OK instead of Replace, you lose the change you made.

When you change the translator for an extension, your project marks all files with that extension for recompilation.

Adding an extension

To add a new extension to the list, follow these steps:

1. Enter the extension in the File Extension box. It must begin with a period (.).

- Choose a translator from the **Translator** menu.

2. Click the Add Button.

Deleting an extension

To remove an extension/translator pair from the list, follow these steps:

1. Select an extension pair in the list.
2. Click the Delete button.

Note

Before you delete an extension, make sure that none of the files in your project uses that extension. You cannot edit or compile a file with an unknown extension.

Changing Your Project Window View

The THINK Project Manager lets you choose how much information is shown in your project windows. You can choose the project components that you see and whether segment numbers are always displayed.

Viewing your project's components

A project contains up to four object components for each source file or library. (Not all project types use all four components.)

This…	Contains…
CODE	The executable code for the project.
DATA	The global and static variables.
JUMP	The jump table, which multi-segment projects use to determine the address of a routine.
STRS	String literals, in applications that have the "Separate STRS" option on. Most projects do not have a STRS component.

Table 8-2 Project components

To examine the size of each component in your project, go to the Project Window page of the **THINK Project Manager options** dialog, shown in Figure 8-22. These four options let you choose the

components that are displayed in the project window: Show CODE size, Show DATA size, Show JUMP size, and Show STRS size.

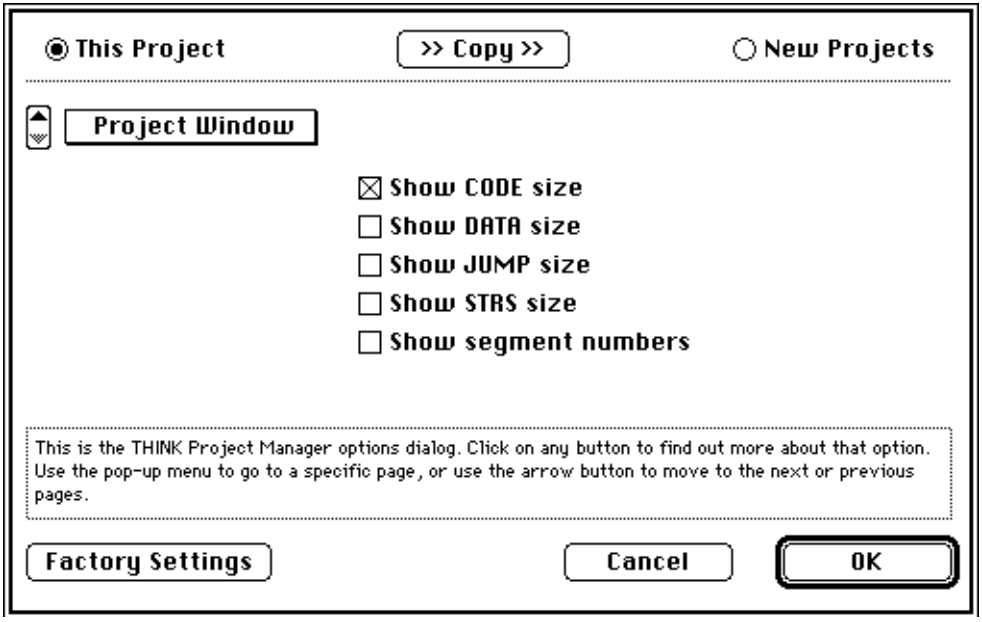
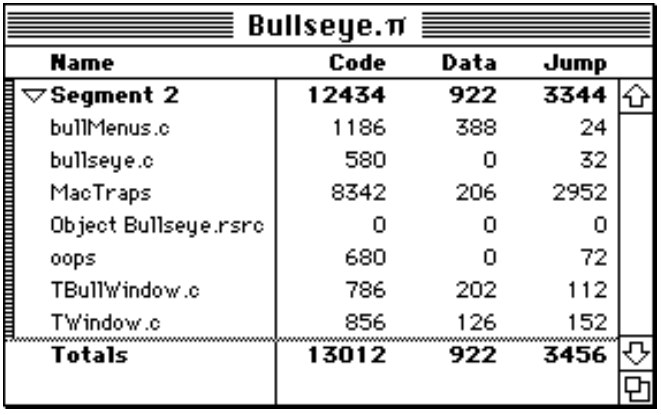


Figure 8-22 Project Window options

If you choose the options for showing the code, data, and jump components, the project window looks like the one shown in Figure 8-23.



Name	Code	Data	Jump
Segment 2	12434	922	3344
bullMenus.c	1186	388	24
bullseye.c	580	0	32
MacTraps	8342	206	2952
Object Bullseye.rsrc	0	0	0
oops	680	0	72
TBullWindow.c	786	202	112
TWindow.c	856	126	152
Totals	13012	922	3456

Figure 8-23 Project window with Code, Data, and Jump columns

The numbers after a file show what that file contributes to a project. Those in the segment line show the totals for that segment and those in the total line show the totals for the whole project.

The columns always appear in the order Code, Data, Jump, and Strs, no matter the order in which you select the options.

Viewing segment numbers

For more information on how to name segments, see "Changing a segment's name and attributes" earlier in this chapter.

The THINK Project Manager lets you choose whether you see the segment numbers for named segments. Go to the Project Window page of the **THINK Project Manager Options** dialog, shown in Figure 8-22. To see segment numbers for named segments, select Show segment numbers. The numbers appear in brackets after the segment names, as in Figure 8-24.

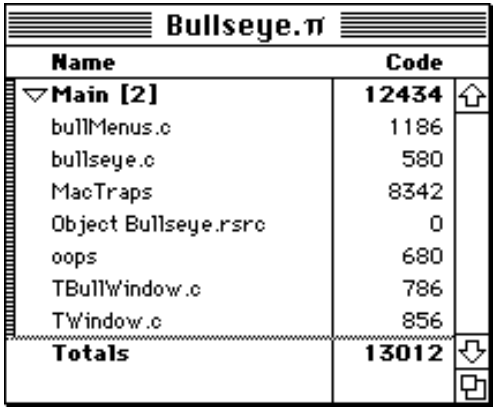


Figure 8-24 Project window with segment numbers

The segment numbers for unnamed segments are always displayed as substitutes for names.

Note

The first segment shown in the project window is segment 2. Segments 0 and 1 are reserved for the run-time and jump table loader.

Project Limits

The maximum size of a project is 16MB. The THINK Project Manager will not compile files once your project reaches that limit. Here are some ways to save space in your project:

- Don't save debugger information for some files, described in "Turning the debugger on" in Chapter 12, "Debugging Programs."
- Save debugging information outside your project, described in "Store debug info separately" in Chapter 12, "Debugging Programs."

Depending on the kind of project you build, there are different limits on the sizes of its components, shown in Table 8-3. If you exceed any of these limits, you see an error at link time.

If project is a(n)…	The limits are…
Application	CODE: 32K per segment DATA: Unlimited per project, 32K per file, with Far DATA (32K per project without Far DATA) JUMP: 256K per project, 32K per file, with Far CODE (32K per project without Far CODE) STRS: Unlimited per project, 32K per file, if "Separate STRS" is on. (No STRS if "Separate STRS" is off. Only THINK C uses STRS.)
Single-segment desk accessory or device driver	CODE: 32K per project DATA: 32K per project JUMP: Not used
Multi-segment desk accessory or device driver	CODE: 32K per segment DATA + 32K per project JUMP:

Table 8-3 Project component limits

If project is a(n)…	The limits are…
Single-segment code resource	CODE + 32K per project DATA: JUMP: Not used
Multi-segment code resource	Segment with `main()`: CODE for segment + DATA for project + JUMP for project: 32K Other segments: CODE:32K

Table 8-3 Project component limits *(Continued)*

THINK C and Symantec C++ generate an 8-byte jump table entry for each function that is not declared `static` and for each function whose address is taken. The jump table built by the **Build** command may be smaller. The entry for each function that is only referenced within a single segment is removed.